ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

1η Σειρά Ασκήσεων

ΔΗΜΗΤΡΙΑΔΗΣ ΓΕΩΡΓΙΟΣ: ΑΜ – 5209
ΔΗΜΗΤΡΙΟΥ ΑΡΙΣΤΟΤΕΛΗΣ: ΑΜ - 5211

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Πίνακας περιεχομένων

Προαπαιτούμενες ενέργειες για την εκτέλεση του κώδικα

Για την ορθή διεκπεραίωση του κώδικα χρειάστηκε να προσθέσουμε τις παρακάτω βιβλιοθήκες:

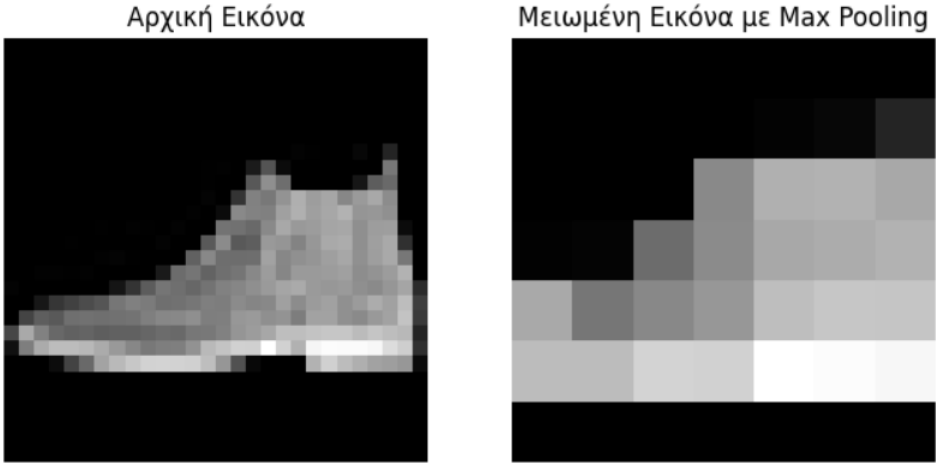
- import numpy as np
- import tensorflow as tf
- from tensorflow.keras.datasets import fashion_mnist
- from tensorflow.keras.layers import MaxPooling2D
- import matplotlib.pyplot as plt
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.metrics import confusion_matrix
- import seaborn as sns
- from sklearn.tree import DecisionTreeClassifier, plot_tree
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.svm import SVC
- from sklearn.preprocessing import StandardScaler
- from sklearn.metrics import accuracy_score
- from sklearn.model_selection import GridSearchCV
- from tensorflow.keras.models import Sequential
- from tensorflow.keras.layers import Dense, LeakyReLU
- from tensorflow.keras.optimizers import Adam
- from tensorflow.keras.layers import Conv2D
- from tensorflow.keras.layers import Flatten
- from tensorflow.keras.layers import Dropout
- from keras.layers import Input, Dense, LeakyReLU
- from keras.models import Model

1η Άσκηση

Α΄ Ερώτημα

- Φόρτωση των δεδομένων Fashion MNIST: Αρχικά φορτώνονται τα δεδομένα εκπαίδευσης και δοκιμής του σετ δεδομένων Fashion MNIST. Αυτά περιλαμβάνουν τις εικόνες και τις αντίστοιχες ετικέτες.
- Κανονικοποίηση των δεδομένων: Τα δεδομένα των εικόνων κανονικοποιούνται διαιρώντας τις τιμές των pixel με το 255.0 για να κλιμακωθούν σε ένα εύρος μεταξύ 0 και 1.
- Προσθήκη διάστασης καναλιού: Οι εικόνες τροποποιούνται για να περιλαμβάνουν μια επιπλέον διάσταση που αντιπροσωπεύει το κανάλι των εικόνων.
- Δημιουργία του max pooling layer: Ορίζεται ένα max pooling layer με μέγεθος παραθύρου 4x4.
- Εφαρμογή του max pooling: Το max pooling layer εφαρμόζεται στα δεδομένα εκπαίδευσης και δοκιμής, μειώνοντας έτσι τις διαστάσεις των εικόνων.
- Εκτύπωση των νέων διαστάσεων: Οι νέες διαστάσεις των δεδομένων εκπαίδευσης και δοκιμής εκτυπώνονται για επαλήθευση της μείωσης των διαστάσεων.
- Επιλογή και εφαρμογή max pooling σε μια εικόνα δοκιμής: Επιλέγεται μια τυχαία εικόνα από τα δεδομένα δοκιμής και εφαρμόζεται σε αυτή max pooling.
- Εμφάνιση της αρχικής και μειωμένης εικόνας: Η αρχική εικόνα και η εικόνα μετά την εφαρμογή του max pooling εμφανίζονται δίπλα-δίπλα για σύγκριση.

```
Νέες διαστάσεις δεδομένων εκπαίδευσης: (60000, 7, 7, 1)
Νέες διαστάσεις δεδομένων ελέγχου: (10000, 7, 7, 1)
```



Β΄ Ερώτημα

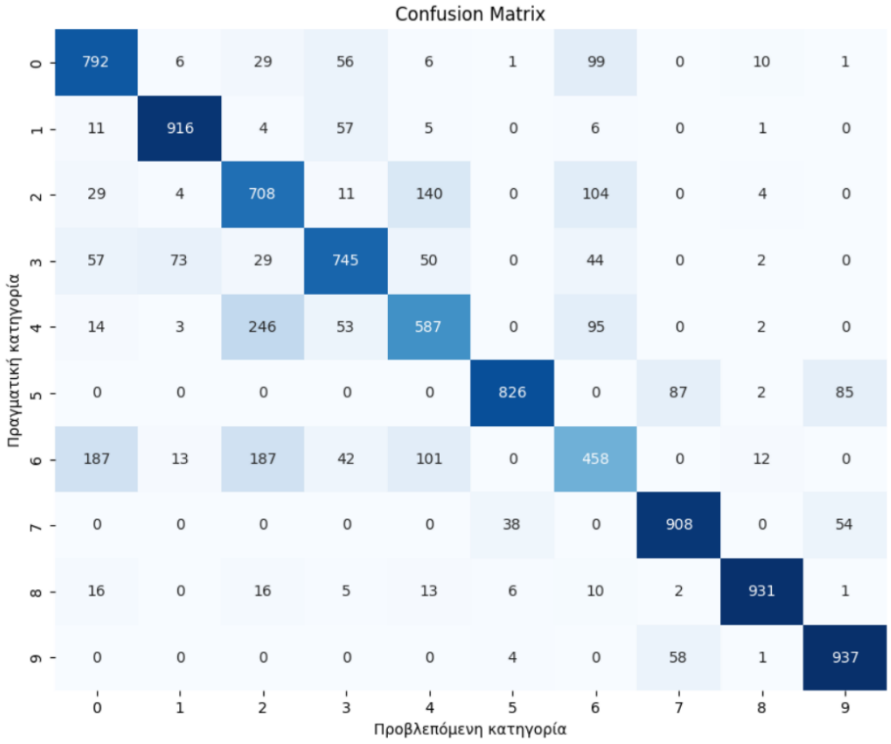
- Δεδομένα Εκπαίδευσης και Ελέγχου: Τα δεδομένα εκπαίδευσης (X_train_flattened) και δοκιμής (X_test_flattened) αναδιαμορφώνονται (flatten) από τις αρχικές τους πολυδιάστατες μορφές, που έχουν προκύψει από την εφαρμογή του max pooling, σε δισδιάστατες δομές για να είναι συμβατά με τον ταξινομητή. Αυτό γίνεται χρησιμοποιώντας την εντολή .reshape(), όπου τα αρχικά δεδομένα αναδιαμορφώνονται ώστε να έχουν μια διάσταση που αντιπροσωπεύει τον αριθμό των δειγμάτων (εικόνων) και μια δεύτερη διάσταση που αντιπροσωπεύει την επίπεδη έκδοση της εικόνας.
- Εκπαίδευση του Ταξινομητή KNN: Ο κώδικας δημιουργεί έναν ταξινομητή KNN με n_neighbors, δηλαδή χρησιμοποιεί τις τρεις πιο κοντινές γειτονιές για την ταξινόμηση ενός νέου δείγματος. Ο ταξινομητής εκπαιδεύεται με τα δεδομένα εκπαίδευσης (X_train_flattened) και τις αντίστοιχες ετικέτες (y_train) χρησιμοποιώντας την μέθοδο .fit().
- Πρόβλεψη με τον Ταξινομητή: Μετά την εκπαίδευση του ταξινομητή, χρησιμοποιείται για να προβλέψει τις ετικέτες (y_pred) των δεδομένων ελέγχου (X_test_flattened). Αυτό επιτυγχάνεται με την κλήση της μεθόδου .predict() στον ταξινομητή.
- Υπολογισμός Ακρίβειας: Ο κώδικας υπολογίζει την ακρίβεια των προβλέψεων μετρώντας το ποσοστό των σωστών προβλέψεων στα δεδομένα ελέγχου. Αυτό γίνεται υπολογίζοντας τη μέση τιμή (np.mean) των περιπτώσεων όπου οι προβλέψεις (y_pred) αντιστοιχούν στις πραγματικές ετικέτες (y_test), και το αποτέλεσμα εκτυπώνεται.

```
Ακρίβεια: 0.7808
```

Στην συνέχεια κάνουμε μια καλύτερη αναπαράσταση με confusion matrix:

- Πρόβλεψη για τα Δεδομένα Ελέγχου: Ο κώδικας χρησιμοποιεί τον εκπαιδευμένο ταξινομητή K-Nearest Neighbors (KNN) για να προβλέψει τις ετικέτες (y_pred) των δεδομένων ελέγχου (X_test_flattened). Αυτό επιτυγχάνεται με την κλήση της μεθόδου .predict() στον ταξινομητή.
- Υπολογισμός του Confusion Matrix: Μετά την πρόβλεψη, ο κώδικας υπολογίζει το confusion matrix (cm) για τα δεδομένα ελέγχου και τις προβλέψεις. Το confusion matrix είναι ένας πίνακας που δείχνει πόσα δείγματα ταξινομήθηκαν σωστά ή λανθασμένα για κάθε κατηγορία.
- Εμφάνιση του Confusion Matrix: Ο κώδικας χρησιμοποιεί τη βιβλιοθήκη Seaborn για να εμφανίσει το confusion matrix ως heatmap. Στη heatmap,

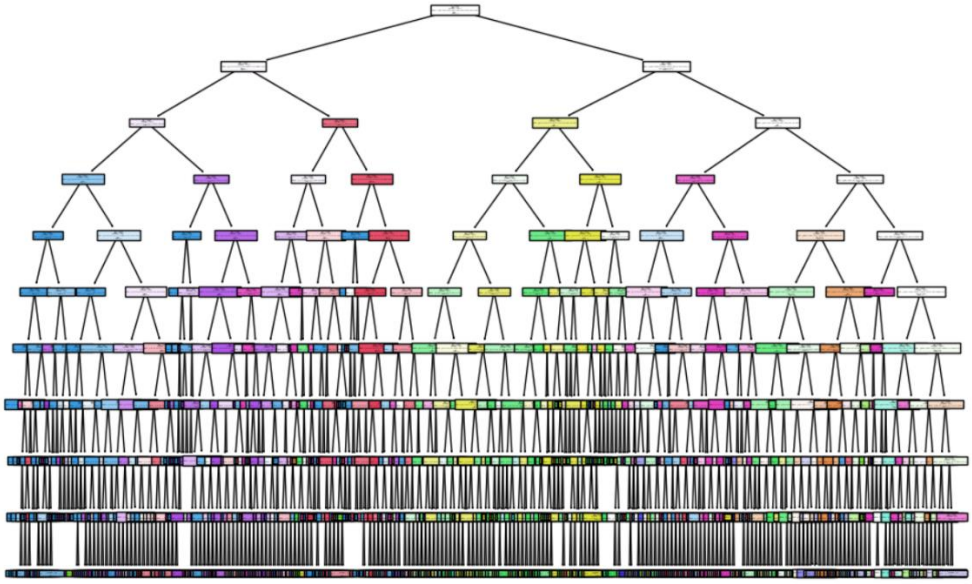
τα τετράγωνα δείχνουν τον αριθμό των δειγμάτων για κάθε συνδυασμό πραγματικής και προβλεπόμενης κατηγορίας. Τα χρώματα αντιπροσωπεύουν την κλίμακα των τιμών, με πιο σκούρα χρώματα να αντιπροσωπεύουν υψηλότερους αριθμούς.



Γ΄ Ερώτημα

- Δημιουργία και Εκπαίδευση Decision Tree: Ο κώδικας δημιουργεί ένα Decision Tree (δέντρο απόφασης) με μέγιστο βάθος (max_depth) 10. Το Decision Tree είναι ένας ταξινομητής που χωρίζει τα δεδομένα σε διάφορες κατηγορίες βασιζόμενο σε ερωτήματα που αφορούν τις τιμές των χαρακτηριστικών. Ο ταξινομητής εκπαιδεύεται με τα δεδομένα εκπαίδευσης (X_train_flattened) και τις αντίστοιχες ετικέτες (y_train) χρησιμοποιώντας τη μέθοδο .fit().
- Plot του Δέντρου Απόφασης: Μετά την εκπαίδευση, ο κώδικας εμφανίζει το Decision Tree χρησιμοποιώντας τη βιβλιοθήκη matplotlib. Αυτό επιτρέπει την οπτικοποίηση της δομής του δέντρου, συμπεριλαμβανομένων των κόμβων, των διακλαδώσεων, και των φύλλων. Το δέντρο σχεδιάζεται με χρώματα που αντιπροσωπεύουν τις κατηγορίες (ειδικότερα, με χρώματα γεμισμένα).

- Δημιουργία και Εκπαίδευση Random Forest: Ο κώδικας δημιουργεί ένα Random Forest (τυχαίο δάσος) με 100 δέντρα (n_estimators=100). Το Random Forest είναι ένας ταξινομητής που συνδυάζει πολλά δέντρα απόφασης για να βελτιώσει την ακρίβεια και τη σταθερότητα των προβλέψεων. Ο ταξινομητής εκπαιδεύεται με τα δεδομένα εκπαίδευσης (X_train_flattened) και τις αντίστοιχες ετικέτες (y_train) χρησιμοποιώντας τη μέθοδο .fit().



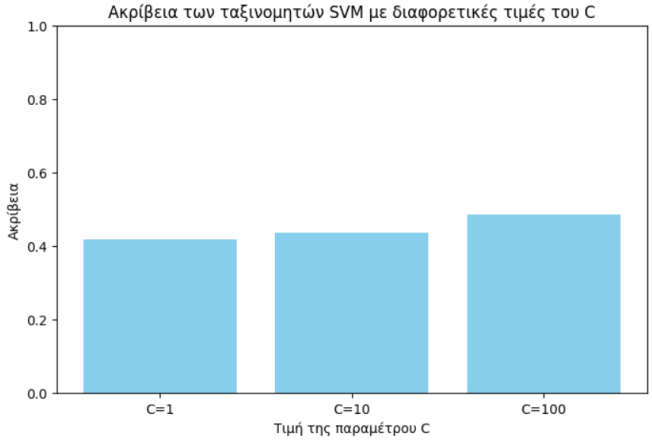
Δ΄ Ερώτημα

(I)

- Κανονικοποίηση των Δεδομένων: Τα δεδομένα εκπαίδευσης (X_train_flattened) και ελέγχου (X_test_flattened) κανονικοποιούνται με τη χρήση του StandardScaler. Αυτό σημαίνει ότι οι τιμές των δεδομένων προσαρμόζονται για να έχουν μέση τιμή 0 και τυπική απόκλιση 1.
- Δημιουργία και Εκπαίδευση των Ταξινομητών SVM: Ταξινομητές Support Vector Machine (SVM) δημιουργούνται και εκπαιδεύονται με διαφορετικές τιμές της παραμέτρου C (C_values = [1, 10, 100]). Η παράμετρος C καθορίζει το επίπεδο της τακτικοποίησης που θα χρησιμοποιηθεί στον ταξινομητή. Χρησιμοποιείται ένας γραμμικός πυρήνας (kernel='linear') και ορίζεται μέγιστος αριθμός επαναλήψεων (max_iter=500) για την εκπαίδευση του ταξινομητή.

- Πρόβλεψη στα Δεδομένα Ελέγχου: Οι εκπαιδευμένοι ταξινομητές χρησιμοποιούνται για να κάνουν προβλέψεις (y_pred) στα κανονικοποιημένα δεδομένα ελέγχου (X_test_scaled).
- Υπολογισμός Ακρίβειας: Η ακρίβεια των ταξινομητών υπολογίζεται μετρώντας το ποσοστό των σωστών προβλέψεων (accuracy) και εκτυπώνεται.
- Σχεδίαση Σχηματικού Τεστ: Σχεδιάζουμε ένα γράφημα τύπου bar που δείχνει την ακρίβεια των ταξινομητών SVM με διαφορετικές τιμές της παραμέτρου C. Αυτό επιτρέπει τη σύγκριση των επιδόσεων των ταξινομητών και την οπτικοποίηση του πώς η τιμή της παραμέτρου C επηρεάζει την ακρίβεια.

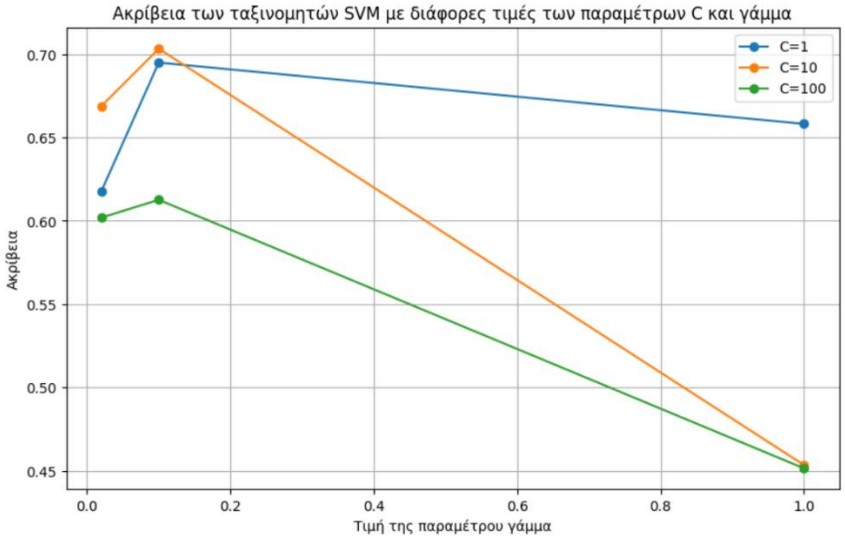
```
C:\Users\georg\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\svm\_base.py:297: ConvergenceWarning: Solver terminated early (max_iter=500).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(
SVM με C=1: Ακρίβεια = 0.42
C:\Users\georg\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\svm\_base.py:297: ConvergenceWarning: Solver terminated early (max_iter=500).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(
SVM με C=10: Ακρίβεια = 0.44
C:\Users\georg\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\svm\_base.py:297: ConvergenceWarning: Solver terminated early (max_iter=500).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(
SVM με C=100: Ακρίβεια = 0.49
```



(II)

- Διερεύνηση Παραμέτρων: Οι τιμές των παραμέτρων C και gamma αποθηκεύονται στις λίστες C_values και gamma_value αντίστοιχα. Ο κώδικας εξερευνά τις επιδόσεις του ταξινομητή για όλες τις δυνατές συνδυασμένες τιμές των C και gamma.
- Δημιουργία του Ταξινομητή SVM: Για κάθε συνδυασμό τιμών C και gamma, ο κώδικας δημιουργεί έναν ταξινομητή SVM με χρήση του πυρήνα RBF (kernel='rbf') και των αντίστοιχων τιμών των C και gamma. Η παράμετρος max_iter=500 περιορίζει τον αριθμό των επαναλήψεων κατά την εκπαίδευση.

- Εκπαίδευση του Ταξινομητή: Ο ταξινομητής εκπαιδεύεται χρησιμοποιώντας τα κανονικοποιημένα δεδομένα εκπαίδευσης (X_train_scaled) και τις αντίστοιχες ετικέτες (y_train).
- Πρόβλεψη και Αξιολόγηση: Οι προβλέψεις (y_pred) γίνονται στα κανονικοποιημένα δεδομένα ελέγχου (X_test_scaled). Η ακρίβεια (accuracy) υπολογίζεται με βάση τις προβλέψεις και τις πραγματικές ετικέτες (y_test).
- Εμφάνιση Αποτελεσμάτων: Οι ακρίβειες αποθηκεύονται στη λίστα accuracies για κάθε συνδυασμό των τιμών C και gamma. Τα αποτελέσματα παρουσιάζονται σε γράφημα, όπου η ακρίβεια για κάθε τιμή C σχεδιάζεται έναντι των τιμών gamma.



Ε΄ Ερώτημα

- Ορισμός του Νευρωνικού Δικτύου: Το νευρωνικό δίκτυο περιλαμβάνει ένα επίπεδο εισόδου (input_layer) με σχήμα (49,), τρία κρυφά επίπεδα (hidden_layer1, hidden_layer2, hidden_layer3) με 100, 100, και 50 νευρώνες αντίστοιχα, και ένα επίπεδο εξόδου (output_layer) με 10 νευρώνες (αντιπροσωπεύοντας τις 10 κατηγορίες Fashion MNIST). Τα κρυφά επίπεδα χρησιμοποιούν λειτουργία ενεργοποίησης Leaky ReLU (LeakyReLU(alpha=0.1)) και το επίπεδο εξόδου χρησιμοποιεί λειτουργία ενεργοποίησης softmax (activation='softmax').

- Δημιουργία του Μοντέλου: Το μοντέλο δημιουργείται χρησιμοποιώντας τη συνάρτηση Model από το Keras, ορίζοντας τα επίπεδα εισόδου και εξόδου.

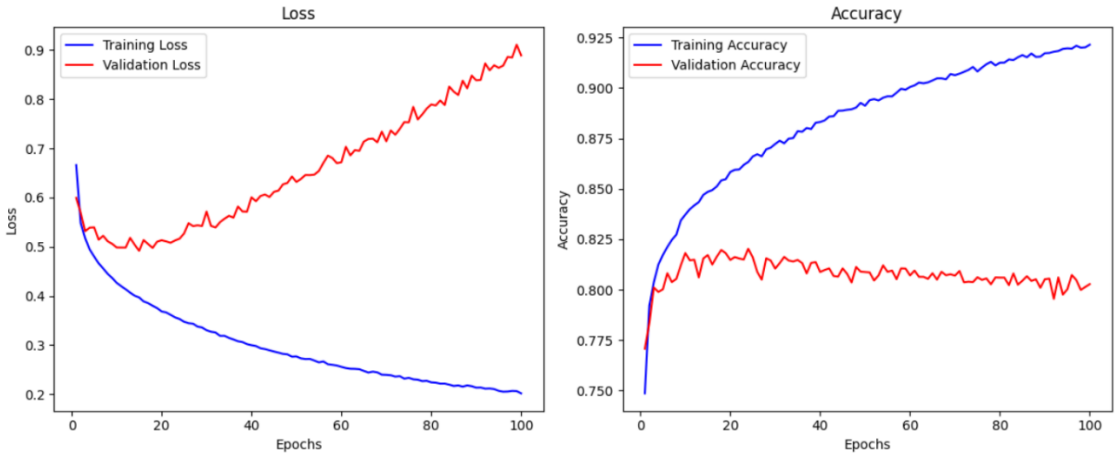
- Σύνολο Εκπαίδευσης: Το μοντέλο συντάσσεται (model.compile) χρησιμοποιώντας τον optimizer Adam και τη συνάρτηση απώλειας sparse_categorical_crossentropy. Μεταξύ των μετρικών, συμπεριλαμβάνεται η ακρίβεια ('accuracy').

- Εκπαίδευση του Μοντέλου: Το μοντέλο εκπαιδεύεται (model.fit) χρησιμοποιώντας τα κανονικοποιημένα δεδομένα εκπαίδευσης (X_train_scaled) και τις αντίστοιχες ετικέτες (y_train) σε 100 εποχές (epochs=100) και μέγεθος παρτίδας (batch_size=50). Οι μετρήσεις αξιολογούνται στα κανονικοποιημένα δεδομένα ελέγχου (validation_data=(X_test_scaled, y_test)).

- Αποθήκευση των Μετρικών: Ο κώδικας αποθηκεύει τις μετρήσεις των τιμών της συνάρτησης απώλειας (loss) και της ακρίβειας (accuracy) τόσο για τα δεδομένα εκπαίδευσης όσο και για τα δεδομένα ελέγχου.

- Εμφάνιση των Μετρήσεων: Δημιουργούμε διαγράμματα που δείχνουν τις τιμές της συνάρτησης απώλειας και της ακρίβειας για κάθε εποχή κατά τη διάρκεια της εκπαίδευσης. Τα διαγράμματα εμφανίζουν τις μετρήσεις τόσο για τα δεδομένα εκπαίδευσης όσο και για τα δεδομένα ελέγχου.

- Εκτύπωση του Πλήθους των Παραμέτρων: Εκτυπώνουμε το συνολικό πλήθος των παραμέτρων του μοντέλου.



Το συνολικό πλήθος των παραμέτρων του μοντέλου είναι: 20660

2η Άσκηση

- Φόρτωση των Δεδομένων: Φορτώνουμε τα δεδομένα εκπαίδευσης (X_train και y_train) και ελέγχου (X_test και y_test) από το σύνολο δεδομένων.
- Κανονικοποίηση των Δεδομένων: Οι τιμές των pixel κανονικοποιούνται σε ένα εύρος [0,1] διαιρώντας τις τιμές με 255.
- Προσθήκη Διάστασης για το Κανάλι Χρώματος: Τα δεδομένα εισόδου έχουν μια επιπλέον διάσταση για το κανάλι χρώματος, καθώς τα δεδομένα είναι ασπρόμαυρα (μία διάσταση).
- Δημιουργία του Μοντέλου: Το μοντέλο είναι βασισμένο στη χρήση του Sequential API και περιλαμβάνει διάφορα επίπεδα:
  1. Δύο συνελικτικά επίπεδα (Conv2D) με 32 και 64 φίλτρα αντίστοιχα, χρησιμοποιώντας kernel_size 3x3 και ενεργοποίηση relu.
  2. Επίπεδο MaxPooling2D για τη μείωση των διαστάσεων των δεδομένων εισόδου.
  3. Ένα επίπεδο Flatten για τη μετουσίωση του χώρου των δεδομένων σε επίπεδο χαρακτηριστικών.
  4. Ένα πλήρως συνδεδεμένο επίπεδο (Dense) με 100 νευρώνες και ενεργοποίηση relu.
  5. Ένα επίπεδο Dropout με ποσοστό 30% για την πρόληψη του overfitting.
  6. Το τελικό επίπεδο Dense έχει 10 νευρώνες με ενεργοποίηση softmax για ταξινόμηση στις 10 κατηγορίες των δεδομένων.
- Σύνθεση του Μοντέλου: Το μοντέλο συντίθεται με τον optimizer Adam, συνάρτηση απώλειας (sparse_categorical_crossentropy), και μετρική (accuracy).
- Εκπαίδευση του Μοντέλου: Το μοντέλο εκπαιδεύεται (fit) για 100 εποχές με batch size 50, χρησιμοποιώντας τα δεδομένα εκπαίδευσης και τα δεδομένα ελέγχου ως επικύρωση (validation_data).
- Αποθήκευση των Μετρικών: Οι τιμές της απώλειας (loss) και του accuracy (accuracy) τόσο για το σύνολο εκπαίδευσης όσο και για το σύνολο ελέγχου αποθηκεύονται στις αντίστοιχες λίστες.
- Απεικόνιση των Μετρικών: Οι μετρήσεις των τιμών της απώλειας και του accuracy κατά τη διάρκεια των εποχών απεικονίζονται σε δύο διαγράμματα, το ένα για την απώλεια και το άλλο για το accuracy.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
conv2d_1 (Conv2D)	(None, 24, 24, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 64)	0
flatten (Flatten)	(None, 9216)	0
dense_4 (Dense)	(None, 100)	921,700
dropout (Dropout)	(None, 100)	0
dense_5 (Dense)	(None, 10)	1,010

```
Epoch 1/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.7859 - loss: 0.6058 - val_accuracy: 0.8927 - val_loss: 0.2960
Epoch 2/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.8941 - loss: 0.2911 - val_accuracy: 0.9037 - val_loss: 0.2568
Epoch 3/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.9123 - loss: 0.2395 - val_accuracy: 0.9143 - val_loss: 0.2375
Epoch 4/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.9263 - loss: 0.1976 - val_accuracy: 0.9148 - val_loss: 0.2313
Epoch 5/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 13s 11ms/step - accuracy: 0.9358 - loss: 0.1692 - val_accuracy: 0.9225 - val_loss: 0.2202
Epoch 6/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.9431 - loss: 0.1504 - val_accuracy: 0.9228 - val_loss: 0.2282
Epoch 7/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.9543 - loss: 0.1228 - val_accuracy: 0.9240 - val_loss: 0.2267
Epoch 8/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.9575 - loss: 0.1095 - val_accuracy: 0.9228 - val_loss: 0.2493
Epoch 9/100
1200/1200 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.9644 - loss: 0.0938 - val_accuracy: 0.9239 - val_loss: 0.2810
Epoch 10/100
```

(Σημείωση: Δεν παρατίθενται όλες τις εποχές για εξοικονόμηση χώρου και χρόνου)

Αποτελέσματα χρόνου:

Συνολικά για την εκτέλεση ολόκληρου του κώδικα απαιτήθηκαν περίπου 25 λεπτά.

Kernel status: Idle

Executed 8 cells

Elapsed time: 1478 seconds